S32 SDK Quick Start Guide

Document Number: S32 SDK QSG
Rev. 1.4, 05/2017

Chapter 1
Introduction

The S32 Software Development Kit (S32 SDK) is an extensive suite of robust hardware interface and hardware abstraction layers, peripheral drivers, RTOS, stacks and middleware designed to simplify and accelerate application development on NXP S32K microcontrollers.

This manual explains how to use the S32 SDK product, including the use with S32 Design Studio, for easier application creation and configuration with graphical user interface.

1.1 System requirements

Hardware	• 1.8 GHz processor • 2 GB of RAM
Operating System	Microsoft® Windows® 7.
Gnu Make	For usage without S32 Design Studio.
Supported Compiler	For usage without S32 Design Studio. See release notes for supported versions.
Disk Space	Approximately 150 MB of free disk space (when installing standalone)

NOTE

The S32 Design Studio software development tools requirements are covered by their own Quick Start Guide. PEx graphical configuration tool can only be used with S32 Design Studio.

1.2 Installing S32 SDK

1.2.1 Bundled in S32 Design Studio

S32 SDK is delivered bundled in the S32 Design Studio. In this case it’s already configured and ready to use.

1.2.2 Standalone

S32 SDK is also delivered through a standalone installer. Using the standalone installer is recommended when using a compiler which is not supported in S32 Design Studio or

when the graphical interface is not required. In this case the installer can configure an existing S32 Design Studio to use the configuration files delivered in the installer.

If the integration with the S32 Design Studio is not needed the path to S32 Design Studio can be left empty – and in this case, only the S32 SDK will be installed and configured.

1.3 Release Notes

Before using the S32 SDK, read the release notes. These notes contain important information about last-minute changes, bug fixes, incompatible elements, or other topics that may not be included in this manual. The product comes with the release notes installed.

1.4 Terminology

The following are some of the terms used in the document.

Table 1-1. Terminology

Term	Description
S32 SDK	Software development kit that provides comprehensive software support for NXP S32 devices. The S32 SDK includes a Hardware Abstraction Layer (HAL) for each peripheral and peripheral drivers built on top of the HAL. S32 SDK also contains the latest available RTOS kernels, a LIN stack and SBC middleware to support rapid development on supported S32K devices.
Processor Expert	Rapid application design tool targeted for NXP microcontrollers providing the following key features: • A Graphical User Interface which allows an application to be specified by the functionality needed. • An application created from Embedded Components encapsulating initialization and functionality of basic elements of embedded systems. • An automatic code generator which creates tested and optimized C code which is tuned to your application needs and the selected NXP device. • A built-in knowledge base, which immediately flags resource conflicts and incorrect settings, so errors are caught early in design cycle allowing you to get to market faster with a higher quality product.

Chapter 2
Working with Projects

This chapter explains how to use the S32 Design Studio to create and work with S32 SDK projects.

A project organizes files and various compiler, linker, and debugger settings associated with the applications or libraries you develop. S32 New Project wizard can be used to create projects that group these files and settings into build and launch configurations.

2.1 Creating and building S32DS project

The New S32DS Project wizard help you to quickly create new projects. The wizard generates a project with placeholder files and default settings (build and launch configurations) for specified target. After the project has been created, you can easily change any default setting to suit your needs.

To create a S32 SDK project using the **New S32DS Project** wizard:

1. Launch the S32 Design Studio. Please refer to S32 Design Studio documentation.

2. Select **File > New > New S32DS Project**, from the IDE menu bar.
   The **New S32DS Project** page of the **New S32DS Project** wizard appears.

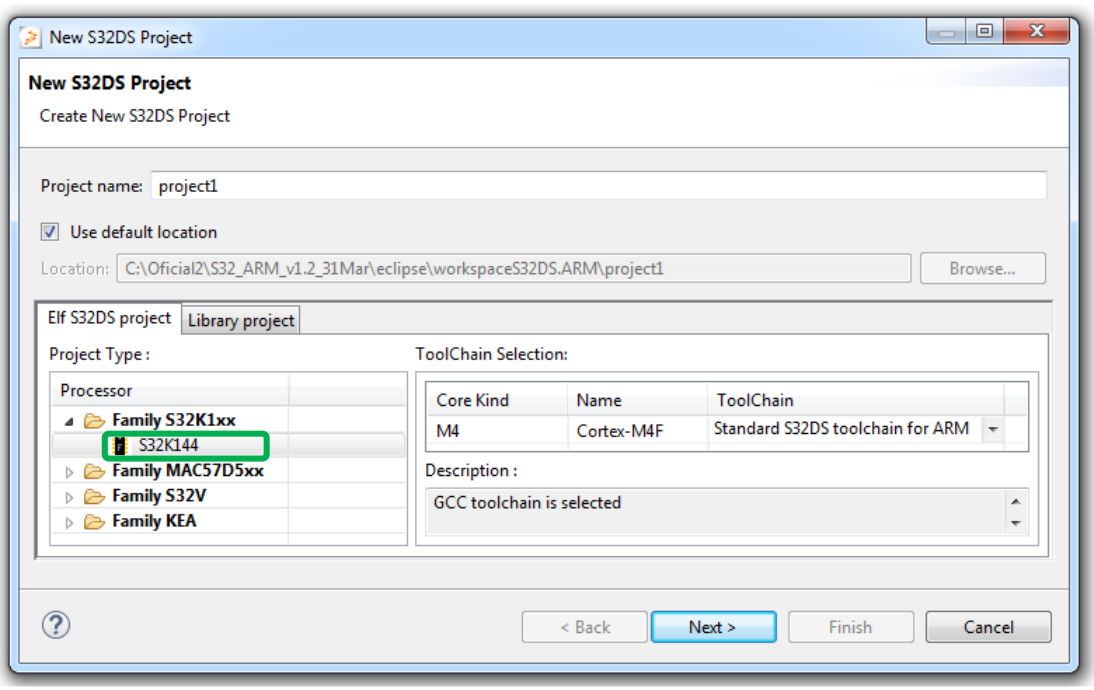

Figure 1 New S32DS Project page

3. Specify a name for the new project. For example, enter the project name as `Project1`.
4. Select the Processor you want to use from **Elf S32DS project** tab
5. Click **Next,** The **New S32DS Project for <CPU_NAME>** page of the **New S32DS Project** wizard appears

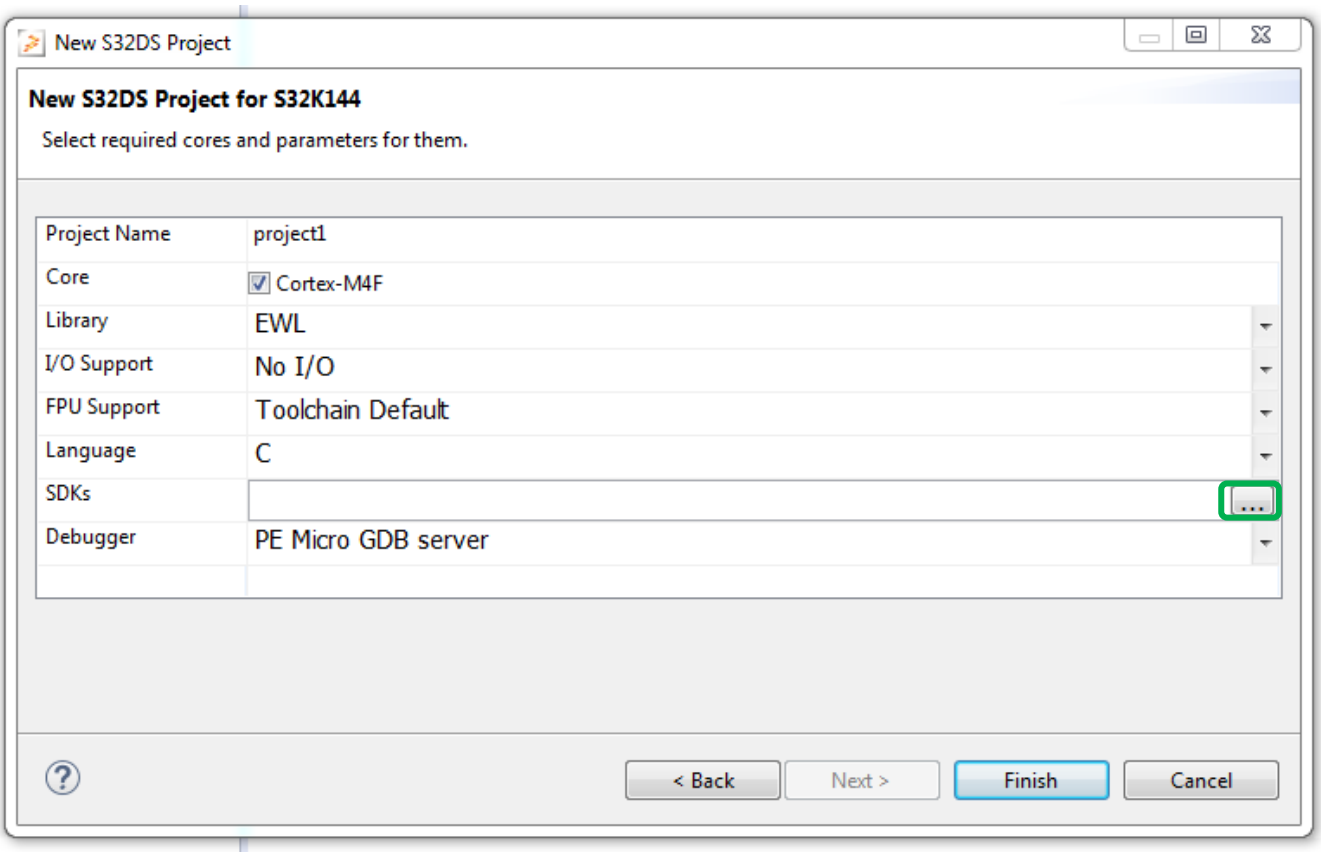

Figure 2 New S32DS Project for <CPU_NAME> page

6. Click on ... button to select from available SDKs**,** The **Select SDK** page of the **New S32DS Project** wizard appears

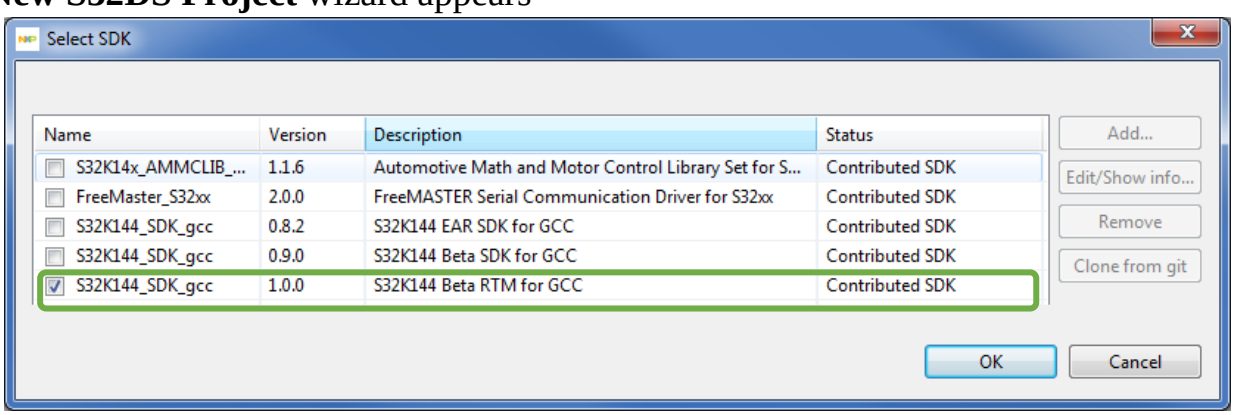

Figure 3 Select SDK page

7. Select the latest version of **SDK**, and then click **OK**.
8. Review the settings and click **Finish,** project should be created and default view should be presented.

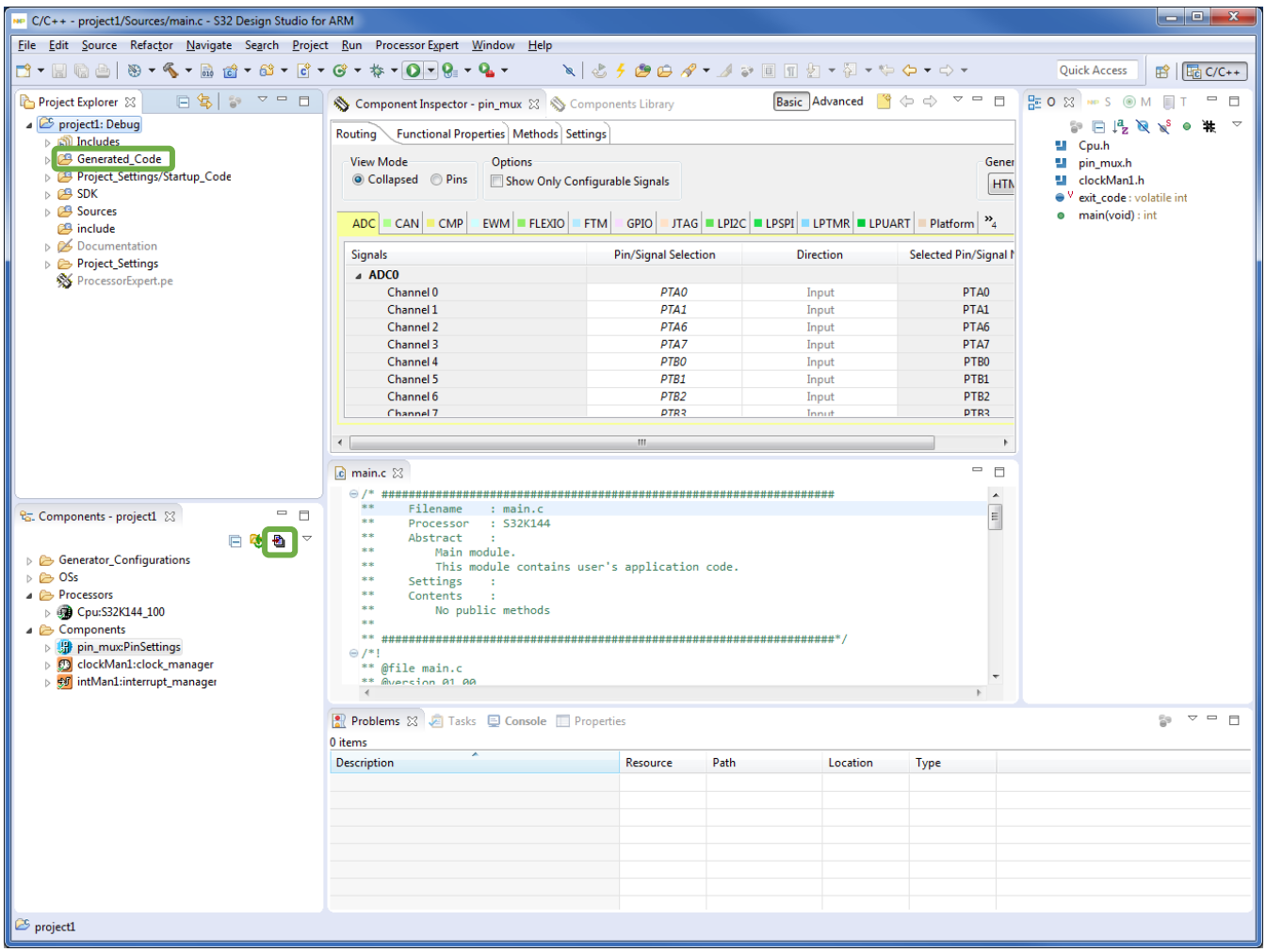

Figure 4 Default view after project creation

9. Processor Expert components can be observed the **Components View**, to add new components open **Component Library**. (which can be opened from toolbar menu Windows -> Show View)
10. Generate the driver configuration by clicking on (or using toolbar menu Project -> Generate Processor Expert Code). The generated configuration will be placed in **Generated_Code** folder in project.

11. Start project build by clicking on , or invoking right click menu on project and selecting Build Project.

2.2 Importing an existing project

This section explains how to import an existing S32 Design Studio SDK project in S32 Design Studio.

To import an existing project:

1. Select **File > Import**, from the IDE menu. **Choose import source** page will appear.

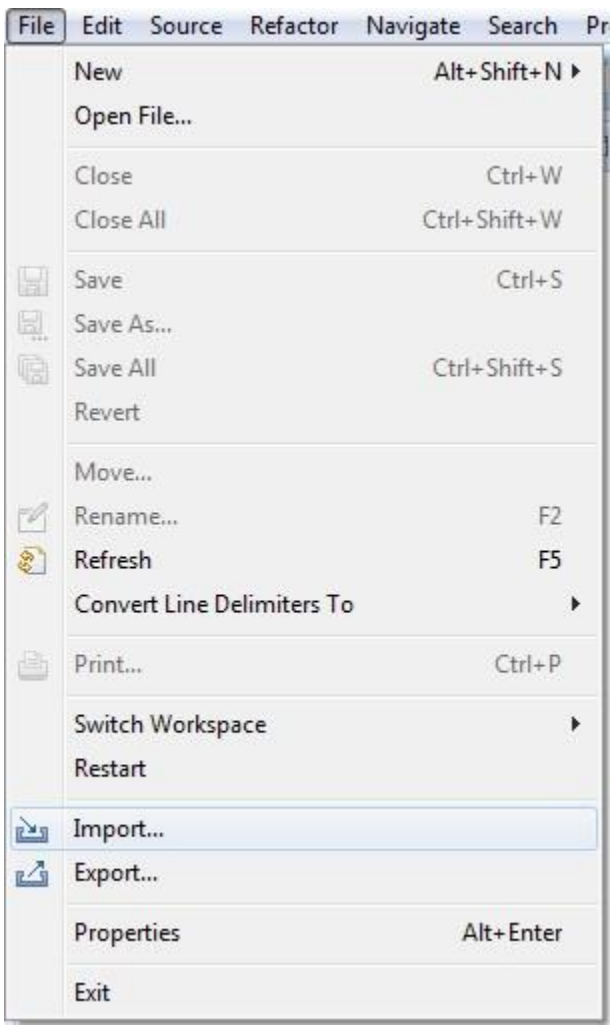

Figure 5 Select import option

2. Expand **General** tree and Select **Existing Projects into Workspace**

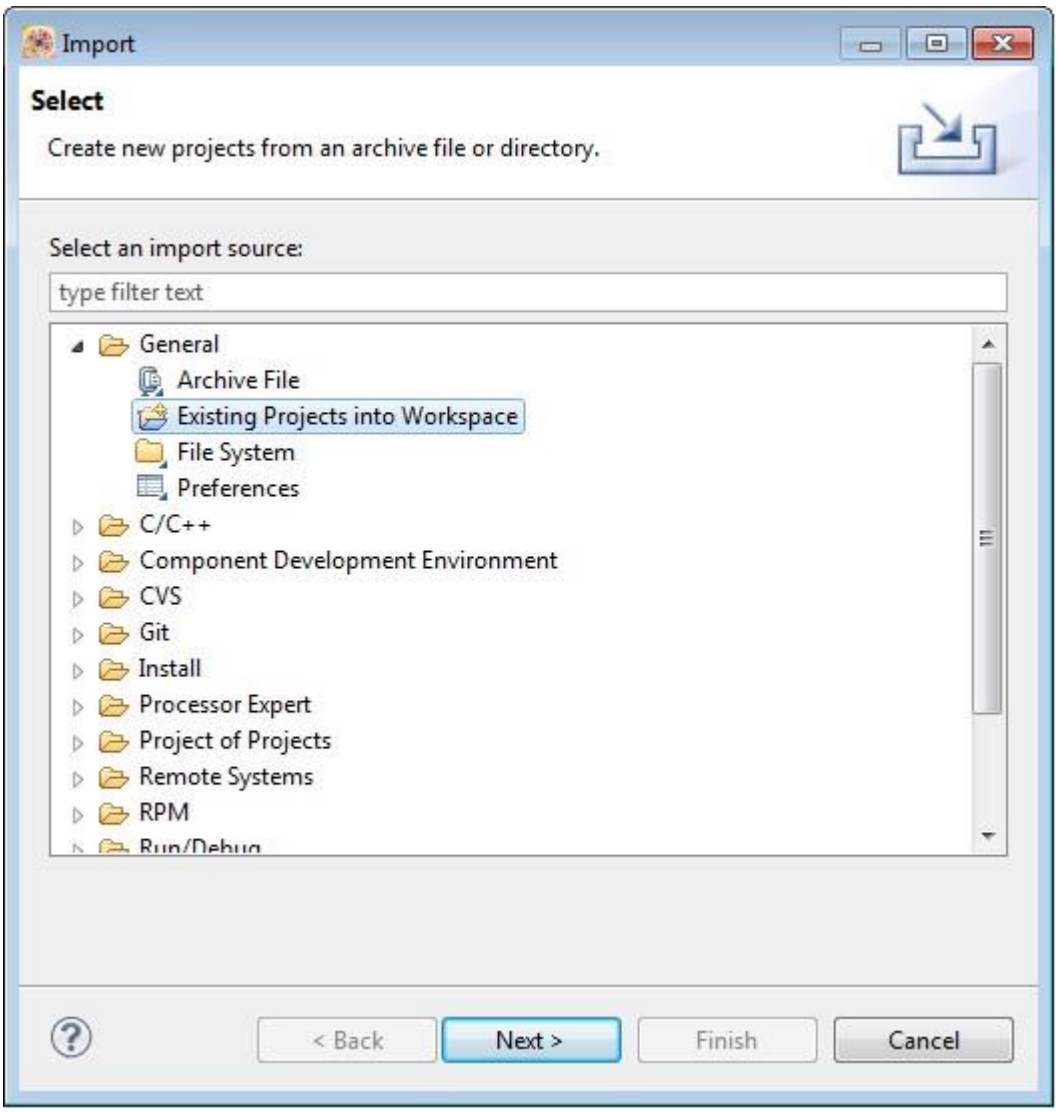

Figure 6 Choose import source

Figure 2-10. Select an import source

3. Click **Next**. The Import projects screen appears.
4. Click **Browse** and select the **example** folder from SDK installation directory to search for an existing Eclipse project.

NOTE

When using the S32 SDK bundled inside S32 Design Studio the example folder is located in **<S32 Design Studio install> \S32DS\S32_SDK_<RELEASE_VERSION>\examples**

5. Select the project you want to import in your Workspace (It is useful that **Copy to Workspace** is selected in this window. In this way the original project will be preserved).
6. Click Finish. The imported project will appear in the Project Explorer view. Similar to **Default view after project creation.**

2.3 Using New Project From Example

S32 Design Studio provides support for accessing the examples from the S32 SDK using the **New Project from Example**. To use this feature follow the next steps:

1. Select **File -> New -> New S32DS Project From Example.** The following window will appear.

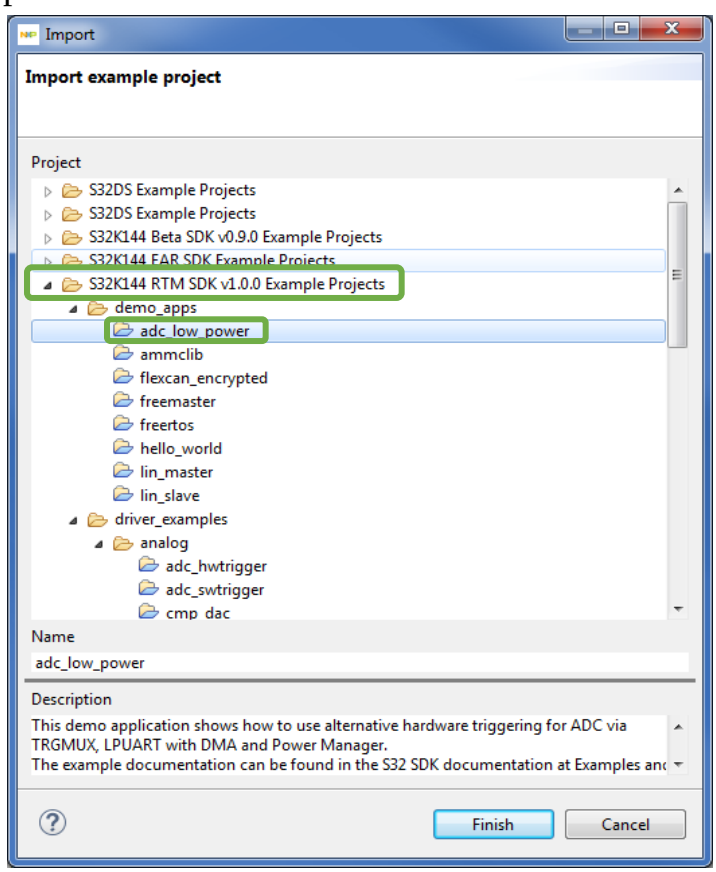

Figure 7 New Project from Example

2. Select the desired project from the release version and click on **Finish**. The project will be copied in your workspace.

2.4 Using a Makefile Project

To build a Makefile project without S32 Design Studio the system needs to have a make utility (GNU version 3.0 and above, or equivalent) and a supported compiler. Please refer to S32 SDK Release Notes for a list of supported compilers and their supported version. Example projects are delivered for all supported compilers.

The makefile projects assumes that the make utility and corresponding compiler are included in path.

1. Open a command line window (Start->Run->cmd.exe)
2. Navigate to project folder - makefile project are locate in example folder.

NOTE

Please check Release Notes or S32 SDK documentation for list of available makefile projects

3. Execute **make all** command. This, depending on the used makefile, will generate one or more elf files.

NOTE

For not relying on System PATH variable the user can temporary modify the **PATH** variable as follows:

```
set PATH=%PATH%;<path to make>;<path to compiler>
```

4. The elf files can be deployed using one of the supported debugger. Please refer to debugger documentation for flashing and debugging the application.

Example:

```
S:\>cd examples\S32K144\demo_apps\blinking_LED\GCC-MKF

S:\examples\S32K144\demo_apps\blinking_LED\GCC-MKF>make all
==============================================================================
Checked for uname, found: MINGW32_NT-6.1
Assuming Unix like environment
==============================================================================
Compiler found in PATH
==============================================================================
Creating directory for object files
==============================================================================
Compiling obj/main.o
==============================================================================
Compiling obj/system_S32K144.o
==============================================================================
Compiling obj/startup.o
==============================================================================
Compiling obj/pcc_hal.o
==============================================================================
Compiling obj/port_hal.o
==============================================================================
Compiling obj/startup_S32K144.o
==============================================================================
Linking to app_ram.elf
==============================================================================
Linking to app_flash.elf
==============================================================================
Build complete!

S:\examples\S32K144\demo_apps\blinking_LED\GCC-MKF>dir
 Volume in drive S is Primary
 Volume Serial Number is BC74-DCFE

 Directory of S:\examples\S32K144\demo_apps\blinking_LED\GCC-MKF

02/09/2017  04:50 PM    <DIR>          .
02/09/2017  04:50 PM    <DIR>          ..
02/09/2017  04:50 PM            56,054 app_flash.elf
02/09/2017  04:50 PM            22,485 app_flash.map
02/09/2017  04:50 PM            55,681 app_ram.elf
02/09/2017  04:50 PM            21,195 app_ram.map
02/07/2017  05:12 PM             4,149 Makefile
02/07/2017  05:12 PM             3,747 Makefile-arm
02/09/2017  04:50 PM    <DIR>          obj
               6 File(s)        163,311 bytes
               3 Dir(s)  282,046,857,216 bytes free
```

2.5 Debugging Projects

When you use the **S32 Design Studio** to create a new project, the wizard sets the debugger settings of the project's launch configurations to default values. You can change these default values based on your requirements.

To debug a project, perform these steps.

1. Launch the IDE and have a project opened and selected.

NOTE

Please ensure that the current perspective is **C/C++** or **Debug**. Current perspective is indicated usually in upper right corner in eclipse window. Use Window -> Open Perspective to change current eclipse perspective. Use Window-> Reset perspective to reset all the Views to their default location and visibility.

2. Click  The **Launch Configuration Selection** dialog appears.
   Alternatively, you can select **Run > Debug Configurations** from the IDE menu bar.

NOTE

The launch/debug configurations are populated with the default settings; these might need to be adjusted to accommodate various configuration. Please refer to S32 Design Studio documentation for Launch configuration settings.

3. Select the launch configuration you want to debug.

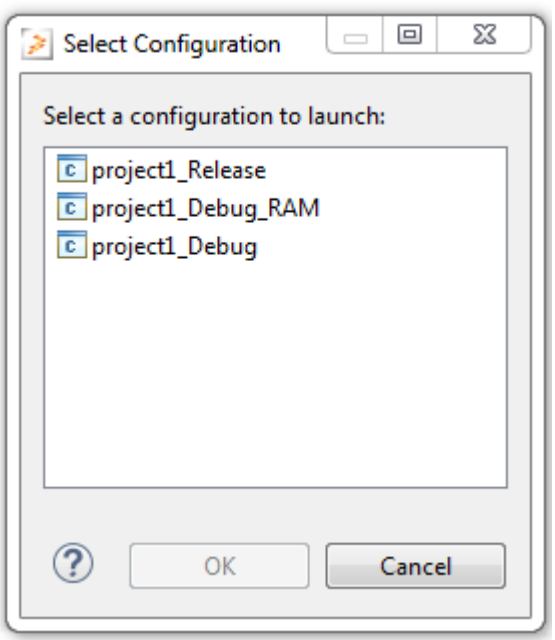

Figure 8 Select launch configuration

4. Click **OK**. The IDE uses the settings in the launch configuration to generate debugging information and initiate communications with the target board.

2.6 Graphical Configuration of Drivers

The simplest way to configure the driver is to use the Processor Expert configurator available in **S32 Design Studio**. Alternatively, the configuration can be written manually in a text editor.

Here are the steps requires to graphically configure a driver:

1. Launch the IDE
2. Open Components View (double click on ProcessorExpert.pe file from project or from toolbar Window->Show View ->Components)
3. If the desired component is not in Components View, open Components Library (Windows -> Show view) and double click on desired component
4. If the need component is in Components View, right click on it and select Inspector from context menu.

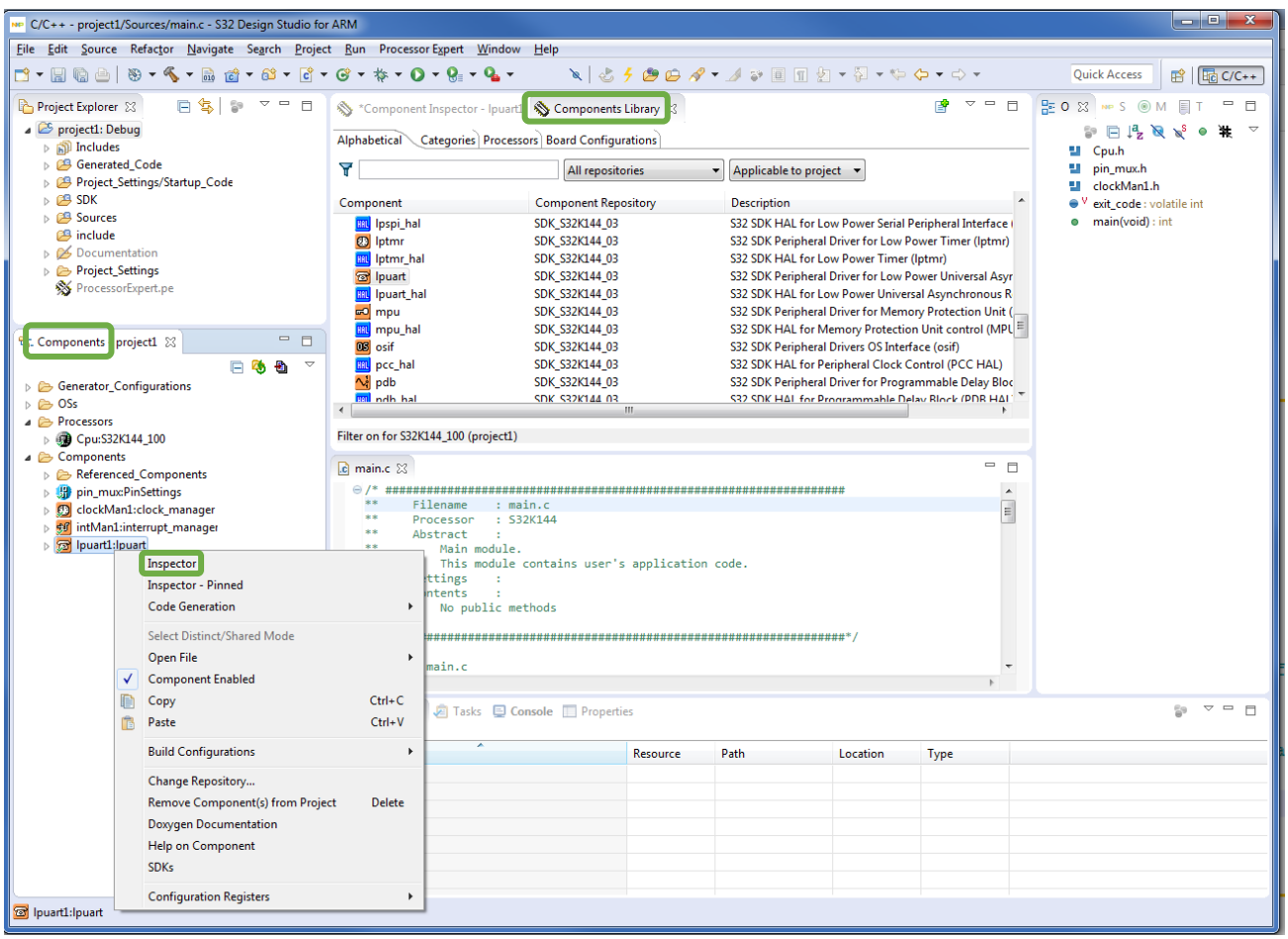

Figure 9 Addition and opening LPUART component

5. After configuring the parameter from UI, click on generate code button ( ), and the configuration will be generated in Generated_Code. In the same time the LPUART driver will be added to project.

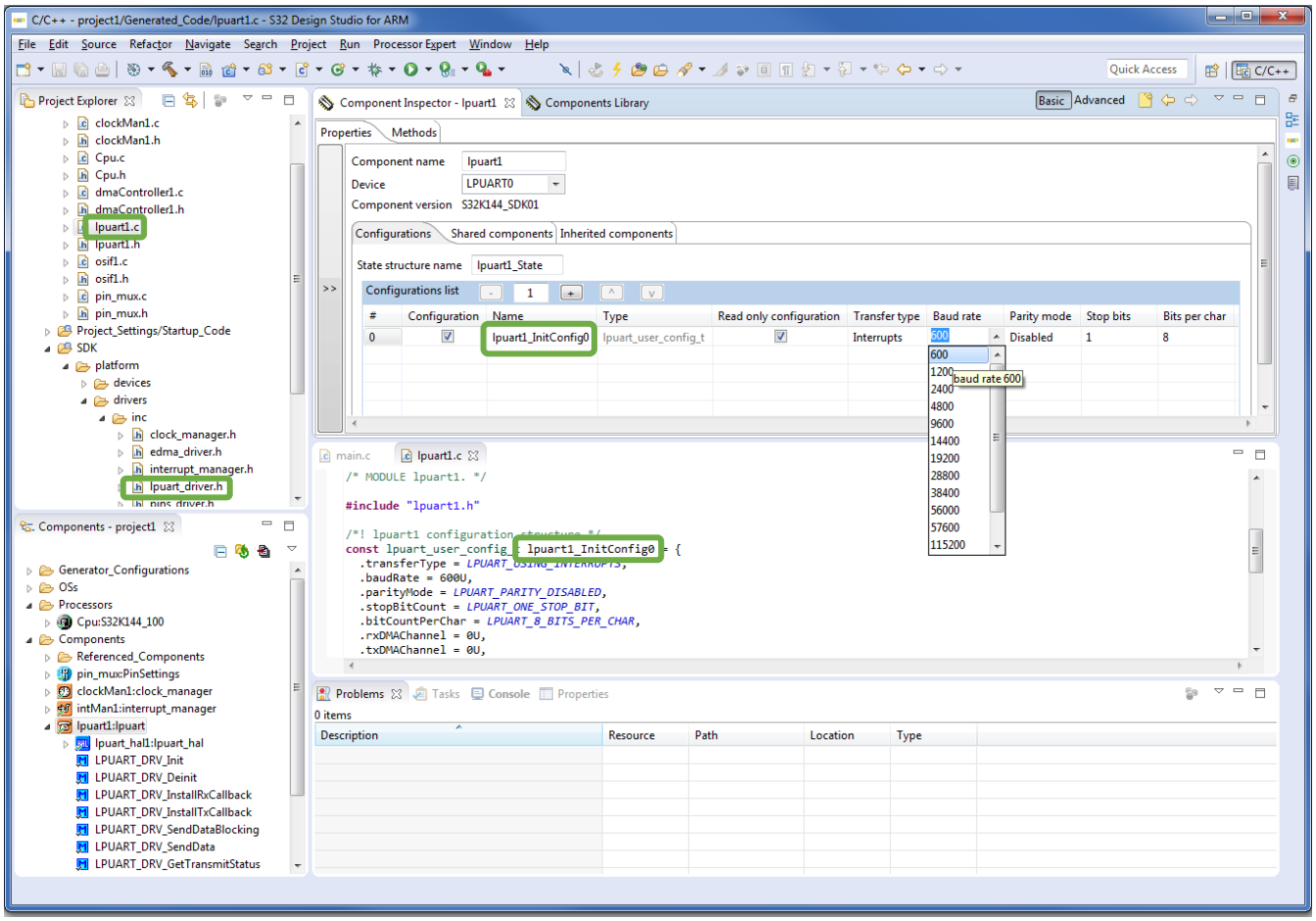

Figure 10 LPUART configuration and driver

6. The generated configuration can be used with driver API, see LPUART example for specific details regarding the configuration structure usage.

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals”, must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.